REMARKS

Reconsideration of the above-identified application in view of the amendments above and the remarks following is respectfully requested.

The office action of July 27, 2007 acted upon claims 1-6, 8-38 and 40-41. Claim 21 was rejected under 35 USC 112, second paragraph. Claims 1-6, 8-11, 13-22, 24-28, 30-31, 33-38 and 40-41 were rejected under 35 USC 103(a). Claim 1 was also rejected under the judicially-created doctrine of obviousness-type double patenting. Claims 12, 23, 29 and 32 were objected to as being dependent upon a rejected base claim. By this response, claims 11-12 and 23 are canceled, without prejudice. New claims 42-456 are introduced. Claims 1-5, 14, 21, 26, 31, 36 and 38 have been amended.

Claims 1-6, 8-10, 13-22, 24-38, and 40-46 are presented in the belief that they recite allowable subject matter.

Interview Summary

An interview was conducted with Examiner Fleurantin on 5 December, 2007. After discussing the various issues, the Examiner considered the arguments made by the Applicant, and asked that Applicant formulate those arguments in the official Response, for further consideration. Various aspects of the interview will be discussed hereinbelow, with regard to the claim rejections.

§ 103 Rejections

The Examiner has rejected claims 1-6, 8-11, 13-22, 24-28, 30-31, 33-38 and 40-41 under § 103(a) as being unpatentable over U.S. Publication No. 2002/0007446 to Stark (henceforth, "Stark") in view of U.S. Patent No. 6,219,662 to Fuh et al. (henceforth, "Fuh"), and further in view of U.S. Patent No. 5,933,104 to Kimura (henceforth, "Kimura"). The Examiner's rejections are respectfully traversed.

With regard to independent claims 1 and 21, the points presented by Applicant include:

1) Examiner's Action had not addressed in the previous Office Action Response, Applicant's explicit assertion that Stark and Fuh are not properly combinable:

> Applicant further argued that Stark '446 and Fuh '662 are not properly combinable. Although, as the Examiner has written, Fuh claims that the transformation module "improves data integrity" [column 8, lines 36-40], Applicant respectfully articulated that the transformation module of Fuh '662 could not improve Stark's data integrity. While Fuh's claim of improved data integrity may be relevant with respect to systems containing "user-defined data" [Fuh '662 column 2, lines 20-25], it is manifest that Stark '446 has no problem of data integrity that Fuh's teachings can solve.

It is well established that obviousness cannot be established by combining pieces of prior art absent some teaching, suggestion or incentive supporting the combination. Here, there is no genuine teaching, suggestion or incentive supporting the combination of Stark and Fuh.

2) Applicant provided further support for the assertion that Stark and Fuh are not properly combinable:

Fuh's teachings explicitly and specifically relate to relational database systems. Like in a "Google" search, a poor choice or representation of the input keys can result in poor "data integrity", such that the results are not what the user intended to retrieve.

By sharp contrast, Stark teaches databases having a one-to-one correspondence between input key and associated data. This is more like looking for an entry in a dictionary. There is, at most, a singular, unique key that matches the input key. Also, once a match is found, there is a unique field of associated data associated with the entry. There is no "guessing" like in Fuh.

In computer science terms, a dictionary is a dynamic set, the set having unique key values (i.e., all the keys are different). Hereinbelow Applicant has included the computer science definition of "dictionary" from the National Institute of Standards and Technology (NIST), which is available at:

http://www.nist.gov/dads/HTML/dictionary.html.

Applicant has further included two pages from Thomas H. Coren et al.: Introduction to Algorithms, MIT Press, a classic, prestigious reference on algorithms, in which the terms "dictionary" and "dynamic set" are defined.

It is manifest that neither Fuh nor Kimura fulfill the basic requirement of "dictionaries" as defined in the art, and their teachings are fundamentally irrelevant to the teachings of Stark, whose method manifestly pertains solely to dynamic sets having unique satellite (associated) data.

It is well established that non-analogous art cannot be used to establish obviousness. See In re Wood, 559 F.2d 1032, 202 USPQ 171, 174 (CCPA 1979). Applicant submits that a person having ordinary skill in the art of ultra high-speed

"dictionaries" would not consider the field of relational database systems to be reasonably pertinent to the particular problem that Applicant was trying to solve.

There is no problem with data integrity in dictionaries, because of the one-to-one correspondence described above. Google-type key searches are not relevant for dictionaries. Consequently, there is no genuine motivation for combining the art of Fuh with the art of Stark.

It is well established that obviousness cannot be established by combining pieces of prior art absent some genuine teaching, suggestion or incentive supporting the combination. Applicant steadfastly maintains that Stark and Fuh are not properly combinable.

3) Examiner's Action combines Stark and Fuh and Kimura. Since Stark and Fuh are not properly combinable, Applicant submits that the additional combination of Kimura is improper.

4) Moreover, the Examiner's cited motivation for combining Kimura with Stark and Fuh appears incomprehensible to Applicant: "to provide number of bits that are included in the offset field depends upon the location of the pattern of data in the sequence of data".

It is well established that obviousness cannot be established by combining pieces of prior art absent some genuine teaching, suggestion or incentive supporting the combination. Applicant steadfastly maintains that the cited motivation for combining Kimura with Stark and Fuh is not a genuine motivation, such that Kimura is not properly combinable with Stark and Fuh.

5) The teachings of Kimura are so far removed from the art of Stark, it is extremely difficult to conceive of any genuine motivation – cited or otherwise – for combining the teachings of Kimura with those of Stark. Applicant submits that the teachings of Kimura have nothing to do with the teachings of Stark, and nothing to do with whatever extremely narrow common denominator that Stark and Fuh may share.

For example: Stark has no offset fields whatsoever. Since Stark is monotonically ordered, an offset field is demonstrably wasteful and absolutely unnecessary. Kimura's art only works with pointers. Stark's art is pointer-less, and this is actually one of the great advantages of Stark. Thus, one skilled in the art would not neutralize the inherent advantages, economies and efficiencies of Stark by trying to introduce the pointer-based technology of Kimura.

6) It is well established that references are not properly combinable if their intended function is destroyed. The use of the slow and wasteful pointer-based technology of Kimura would destroy the intent of Stark to have a method that is extremely quick, robust, and space efficient.

7) Kimura does not perform "a pre-determined transformation of each said key entry to produce said respective coded entry", as recited in the previously submitted independent claims. A key transformation is generally understood in the art to be an operation on a particular key in which a key entry is transformed into another (coded or transformed) key that retains some portion of the original data.

In the instant invention, a significant and useful portion of the data inherent in the original key is retained by the coded or transformed key, even though the coded key has a short length in relation to the full key entry. As further articulated hereinbelow, Kimura does not teach a transformation of key entries, rather, he teaches a log base 2 – based method for reserving a space for pointing to where the data is located. This has nothing to do with key transformations as known in the art and as performed in the context of the present invention.

Thus, Applicant submits that for any one of these reasons, and certainly for the reasons in combination, Kimura is not properly combinable with Stark and Fuh.

With regard to claims 17, 19, 27, and 41:

8) While the Examiner asserted, in the Office Action, that Kimura teaches the log base 2 transformation of key entries disclosed by the instant invention, Applicant argues that a close reading of Kimura shows that Kimura does not teach such a transformation of key entries.

The compression /decompression method of Kimura is an improvement to the LZRW1 method. The LZRW compression method is based on the premise that many blocks of data in the chunk are repetitive. During compression, if a particular block of the chunk's data has been identified as being already compressed, rather than repeating the compression once again and storing once again the compressed data, one can store the previous location and the block length that has already been compressed. This location is indicated as an 'offset' (expressed in number of bytes in relationship to the prior occurrence of the block) in relationship to the current position

within the chunk, and the block's length (expressed also in number of bytes). Since the main memory can be randomly accessed, such a block is instantaneously accessed, thus facilitating on the fly compression/decompression.

Kimura relates to "on the fly" file data compression and decompression. The purpose is efficient file storage on secondary storage (a disk) of a computer.

Kimura's method is based upon keeping the Copy Token fixed in size. This is demonstrated in Fig. 11B, which shows a Copy Token of 16 bits, i.e., fixed in length. This is the improvement over the original LZRW1 algorithm. This is accomplished by the Offset and the Length fields varying in length, but always summing-up to 16 bits.

Therefore, Kimura's method can handle various combinations of Offset/Length pairs, as demonstrated in the table of column 6, lines 15-25. Kimura calls these combinations a "sliding window".

In lines 32-45 in column 7, cited by the Examiner, Kimura discloses how an Offset/Length pair is selected from one of the options appearing in the table. The process is fully described by the flowchart in FIG. 13. This flowchart basically says:

1. If the block which instance has already occurred is two bytes or less, don't bother to replace it by the Copy Token, since the Copy Token is also 2 bytes (16 bits) and no storage savings will be accomplished. In this case the one byte or two byte data is left as-is (literal).
2. If, however, a match is found to be longer than two bytes, the Copy Token selects the longest possible match. Here, the longest possible match is in relationship to the Offset/Length combinations, which options are listed in the table of column 6, lines 15-25.

FIG. 14 is a flowchart detailing the method by which the longest match and eventually an Offset/Length pair are selected, using the following rules:

1. First calculate the offset. This is the difference in between the current pointer position and the location of the previous block's distance. This distance, based upon the page table (column 6), can assume one of the ranges in the table.

   Example: The distance (also called a displacement range) is 751. This corresponds to the 7th displacement range option in the table of column 6, lines 15-25. To represent any number in this displacement range requires 10 bits.

   The number of offset bits required is given by:

$$N^{o}\_of\_Offset\_bits = \log_2 (CP - SB) \qquad (I)$$

   wherein:
   CP – Current Position Pointer
   SB- Start Buffer Pointer, which is the previous location occurrence of the same block.

2. Once the number of bits required for offset is determined, the number of Copy Token bits left for the block's length is given by:

$$N^{o}\_of\_Length\_bits = 16 - N^{o}\_of\_Offset\text{-}bits \quad (II)$$

It is manifest that formula (I) of Kimura relates solely to reserving space for pointing to where the data is located, and has nothing to do with the actual content of the key.

This in no way resembles a key transformation, which is generally understood in the art to be an operation on a particular key in which a key entry is transformed into another (coded or transformed) key that retains some portion of the original data.

In the instant invention, by sharp contrast, a significant and useful portion of the data inherent in the original key is retained by the coded or transformed key, even though the coded key has an extremely short length in relation to the full key entry, i.e., substantially equal to the log-base 2 of the full key entry.

Thus, Applicant argues that a close reading of Kimura shows that Kimura does not teach a transformation of key entries, rather, he teaches a log base 2 – based method for reserving a space for pointing to where the data is located. Consequently, Applicant respectfully submits that the limitations recited by claims 17, 19, 27, and 41 are patentably distinct from the combination of the cited prior art.

With regard to claim 20:

9) Original claim 20 recites a unidirectional transformation. In rejecting claim 20, the Examiner writes that "the limitations of claim 20 are similar to claim 1, therefore, the limitations of claim 20 are rejected in the analysis of claim 1, and this claim is rejected on that basis".

Applicant has carefully reviewed the analysis of claim 1, but fails to find any basis for rejecting the limitation of a "unidirectional transformation" as recited by claim 20.

Moreover, Applicant notes that Kimura performs both compression and decompression, and is therefore bi-directional. Applicant failed to find a unidirectional transformation in the teachings of Kimura. Moreover, such a transformation would appear to run counter to the object of the Kimura invention and to the object of LZRW compression methods in general.

Thus, Applicant respectfully submits that the limitation of a "unidirectional transformation", as recited by claim 20, is patentably distinct from the cited prior art.

While continuing to steadfastly traverse the Examiner's rejections, the Applicant has, in order to expedite the prosecution, chosen to amend independent claims 1 and 21 in order to further clarify and emphasize the crucial distinctions between the method of the present invention and the combined art of Stark, Fuh, and Kimura cited by the Examiner.

Specifically, claims 1 and 21 have been amended to clarify that the limitation "transform each key entry of said key entries into a respective coded entry" is to be construed within the clear context of the instant Specification, i.e., "said respective coded entry containing information corresponding to some information present in said key entry".

As articulated hereinabove, the "key transformation" of Kimura has nothing to do with key transformations, and certainly has nothing to do with key transformations which explicitly retain intrinsic information corresponding to some information present in the key entry.

Support for this limitation is drawn from the Specification, inter alia, from Tables 1-3 and the associated text.

Applicant respectfully submits that claims 1 and 21, and all claims depending therefrom, are patentably distinct from the combination of art cited by the Examiner.

Additional Patentable Features

Unique Matching

Amended claims 2 and 38 now recite the limitation that the match between an input key and a key entry of said key entries "is a unique match". This is a salient feature of the instant invention. This feature is characteristic of "dictionaries", and does not exist in the relational database systems and methods of Fuh and Kimura. While Stark discloses unique matching, the instant invention teaches how to obtain unique matching while using coded, compacted keys that retain only a small portion of the original data.

Support for the amended claims is drawn from the Specification, inter alia, from page 19, lines 6-20, and from page 46, in the "Proof of the Exact Search Algorithm".

Applicant respectfully submits that this limitation makes claims 2 and 38 patentably distinct from the combination of the cited art.

Search Coded Keys, Then Search Full Keys

Amended claims 3 and 36, and new claims 44 and 46, recite the limitation that the performing of the deterministic search in the at least one data structure includes:

> (i) searching said coded entries to identify a potential match between said input key and said key entry of said key entries, and
>
> (ii) searching said key entries to determine whether said potential match is said particular match.

This feature is not taught, nor fairly suggested, by the combination of prior art cited by the Examiner.

In view of this explicit limitation, Applicant respectfully submits that claims 3, 36, 44 and 46 are patentably distinct from the combination of the cited art.

Obviousness-Type Double Patenting Rejections

Claim 1 has been rejected under the judicially created doctrine of obviousness-type double patenting, as being unpatentable over at least one claim of U.S. Patent No. 7,076,602. The Examiner initially refers to claims 1 and 2 of U.S. Patent No. 7,076,602, but subsequently relies on claim 31 of U.S. Patent No. 7,076,602, leading Applicant to understand that there is a mistake in the text of the Examiner's Action. In this response, Applicant assumes that the Examiner meant to refer solely to claim

The Examiner asserts that it would have been obvious "to interchangeably 'storing a plurality of key entries' [sic] to 'performing of each key entry of said plurality of key entries so as to obtain a plurality of coded entries' in order to provide an identification of whether a range boundary is closed or open".

As argued with respect to Kimura above, U.S. Patent No. 7,076,602 does not teach key transformation. The method of U.S. Patent No. 7,076,602 requires range ***validation***. To this end, a range boundary can be represented as closed or open using a '0' or '1'. This in no way resembles a key transformation, which is generally understood in the art to be an operation on a particular key in which a key entry is transformed into another (coded or transformed) key that retains some portion of the original data of that key.

In the instant invention, by sharp contrast, a significant and useful portion of the data inherent in the original key is retained by the coded or transformed key, even

though the coded key has an extremely short length in relation to the full key entry, i.e., substantially equal to the log-base 2 of the full key entry.

Thus, Applicant steadfastly maintains that previously presented claim 1 is indeed patentably distinct from the cited claim of U.S. Patent No. 7,076,602.

Moreover, claim 1 has been amended, without prejudice, and the limitation: "said respective coded entry containing information corresponding to some information present in said key entry" is now more explicitly recited therein. This limitation serves to further distinguish claim 1 from the cited claim of U.S. Patent No. 7,076,602.

Since amended claim 1 is patentably distinct from the claim cited by the Examiner, the Examiner is respectfully requested to reconsider and withdraw the rejection under the judicially created doctrine of obviousness-type double patenting.

§ 112, Second Paragraph Rejections

The Examiner has rejected claim 21 under § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which the Applicant regards as the invention.

Specifically, the Examiner has asserted that the claim is incomplete for omitting the essential element: "wherein a function for performing said pre-determined transformation is substantially independent of specific content of each said key entry of said key entries".

As Applicant argued in the above-referenced interview that although this element did not appear in original independent claim 21, the Examiner found no "gap

between the elements" until the instant Office Action. Nor did the Examiner find a "gap between the elements" with regard to original claims 22-38 depending from claim 21.

Moreover, this "essential element" did not appear in any of the other original independent claims, and again, the Examiner found no "gap between the elements" in those original claims.

In addition, Applicant respectfully maintains that the instant invention is capable of functioning using a function that is dependent on the specific content of the key entries. Rather, one advantage of the instant invention is that the function is independent of the specific content of the key entries. Thus, Applicant respectfully submits that there is no gap in the elements of claim 21, and that claim 21 is free from the deficiencies identified by the Examiner under § 112, second paragraph.

New Claims

Support for new claims 42-46 can be found in the Specification. New independent claim 42 draws support, inter alia, from original claim 12 and the intervening claims. Support for new claims 43, 44 and 46 has been provided hereinabove. New claim 45 draws support, inter alia, from original claim 23 and the intervening claims.

The Examiner had objected to claims 12 and 23 as being based on rejected base claims, but noted that these claims would be allowable if rewritten in independent form including all the limitations of the base claim and any intervening claim.

Thus, Applicant respectfully submits that new claims 42-45 are allowable in their present form.

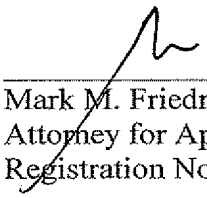
Additional Prior Art References

The Examiner has made of record, and not relied upon, US Patent No. 7,219,184 to Stojancic, but considers the document pertinent to the instant disclosure.

This patent document has been reviewed carefully. Applicant believes that the document does not render the claimed invention unpatentable. In at least some respects, the document tends to indicate the non-obviousness of the present invention.

In view of the above amendments and remarks, it is respectfully submitted that claims 1-6, 8-10, 13-22, 24-38, and 40-46 are in condition for allowance. Prompt notice of allowance is respectfully and earnestly solicited.

Respectfully submitted,



Mark M. Friedman
Attorney for Applicant
Registration No. 33,883

U.S. Telephone Number:
(301)9521011

Date: December 27, 2007

Implementation

(C++, Pascal, and Fortran). Kaz Kylheku's Kazlib (C) implementing dictionary, dynamic hash table, red-black tree, and doubly linked list. Herbert Glarner's Patricia tree (Linoleum) implementation.

Go to the Dictionary of Algorithms and Data Structures home page.

If you have suggestions, corrections, or comments, please get in touch with Paul E. Black.

Entry modified 21 May 2007.
HTML page formatted Mon May 21 08:45:03 2007.

NIST
National Institute of Standards and Technology

dictionary

(data structure)

Definition: An *abstract data type* storing items, or values. A value is accessed by an associated *key*. Basic operations are new, insert, find and delete.

Formal Definition: The operations new(), insert(k, v, D), and find(k, D) may be defined with *axiomatic semantics* as follows.

1. new() returns a dictionary
2. find(k, insert(k, v, D)) = v
3. find(k, insert(j, v, D)) = find(k, D) if $k \neq j$

where k and j are keys, v is a value, and D is a dictionary.

The modifier function delete(k, D) may be defined as follows.

4. delete(k, new()) = new()
5. delete(k, insert(k, v, D)) = delete(k, D)
6. delete(k, insert(j, v, D)) = insert(j, v, delete(k, D)) if $k \neq j$

If we want find to be a *total function*, we could define find(k, new()) using a special value: *fail*. This only changes the return type of find.

7. find(k, new()) = *fail*

Also known as association list, map, property list.

Generalization (I am a kind of ...)
binary relation, *abstract data type*.

Specialization (... is a kind of me.)
associative array.

See also *total order*, *set* Some implementations: *linked list*, *hash table*, *B-tree*, *jump list*, *directed acyclic word graph*.

Note: The terms "association list" and "property list" are used with LISP-like languages and in the area of Artificial Intelligence. These suggest a relatively small number of items, whereas a dictionary may be quite large. Professionals in the Data Management area have specialized semantics for "dictionary" and related terms.

A dictionary defines a binary relation that maps keys to values. The keys of a dictionary are a set.

Contributions by Rob Stewart 16 March 2004.

Author: PEB

Some dynamic sets presuppose that the keys are drawn from a totally ordered set, such as the real numbers, or the set of all words under the usual alphabetic ordering. (A totally ordered set satisfies the trichotomy property, defined on page 31.) A total ordering allows us to define the minimum element of the set, for example, or speak of the next element larger than a given element in a set.

Operations on dynamic sets

Operations on a dynamic set can be grouped into two categories: ***queries***, which simply return information about the set, and ***modifying operations***, which change the set. Here is a list of typical operations. Any specific application will usually require only a few of these to be implemented.

SEARCH(S, k) A query that, given a set S and a key value k, returns a pointer x to an element in S such that $key[x] = k$, or NIL if no such element belongs to S.

INSERT(S, x) A modifying operation that augments the set S with the element pointed to by x. We usually assume that any fields in element x needed by the set implementation have already been initialized.

DELETE(S, x) A modifying operation that, given a pointer x to an element in the set S, removes x from S. (Note that this operation uses a pointer to an element x, not a key value.)

MINIMUM(S) A query on a totally ordered set S that returns the element of S with the smallest key.

MAXIMUM(S) A query on a totally ordered set S that returns the element of S with the largest key.

SUCCESSOR(S, x) A query that, given an element x whose key is from a totally ordered set S, returns the next larger element in S, or NIL if x is the maximum element.

PREDECESSOR(S, x) A query that, given an element x whose key is from a totally ordered set S, returns the next smaller element in S, or NIL if x is the minimum element.

The queries SUCCESSOR and PREDECESSOR are often extended to sets with nondistinct keys. For a set on n keys, the normal presumption is that a call to MINIMUM followed by $n - 1$ calls to SUCCESSOR enumerates the elements in the set in sorted order.

The time taken to execute a set operation is usually measured in terms of the size of the set given as one of its arguments. For example, Chapter 14 describes a data structure that can support any of the operations listed above on a set of size n in time $O(\lg n)$.

Sets are as fundamental to computer science as they are to mathematics. Whereas mathematical sets are unchanging, the sets manipulated by algorithms can grow, shrink, or otherwise change over time. We call such sets ***dynamic***. The next five chapters present some basic techniques for representing finite dynamic sets and manipulating them on a computer.

Algorithms may require several different types of operations to be performed on sets. For example, many algorithms need only the ability to insert elements into, delete elements from, and test membership in a set. A dynamic set that supports these operations is called a ***dictionary***. Other algorithms require more complicated operations. For example, priority queues, which were introduced in Chapter 7 in the context of the heap data structure, support the operations of inserting an element into and extracting the smallest element from a set. Not surprisingly, the best way to implement a dynamic set depends upon the operations that must be supported.

Elements of a dynamic set

In a typical implementation of a dynamic set, each element is represented by an object whose fields can be examined and manipulated if we have a pointer to the object. (Chapter 11 discusses the implementation of objects and pointers in programming environments that do not contain them as basic data types.) Some kinds of dynamic sets assume that one of the object's fields is an identifying ***key*** field. If the keys are all different, we can think of the dynamic set as being a set of key values. The object may contain ***satellite data***, which are carried around in other object fields but are otherwise unused by the set implementation. It may also have fields that are manipulated by the set operations; these fields may contain data or pointers to other objects in the set.